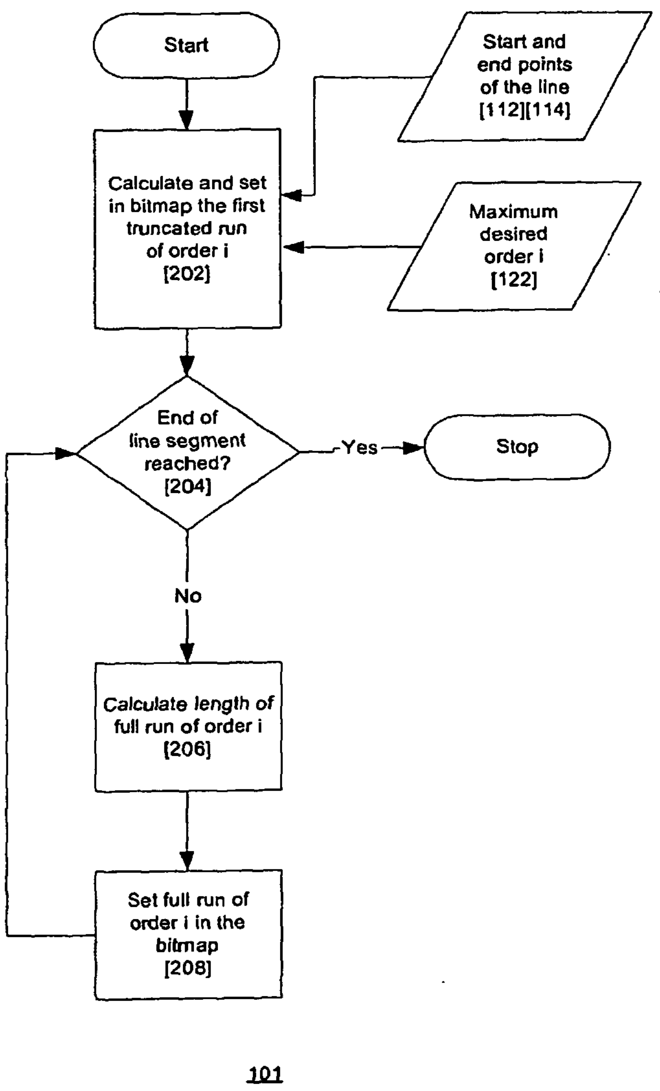

Start
Start and end points of the line [112][114]
Calculate and set in bitmap the first truncated run of order i [202]
Maximum desired order i [122]
End of line segment reached? [204]
Yes
Stop
No
Calculate length of full run of order i [206]
Set full run of order i in the bitmap [208]
101

Fig. 1

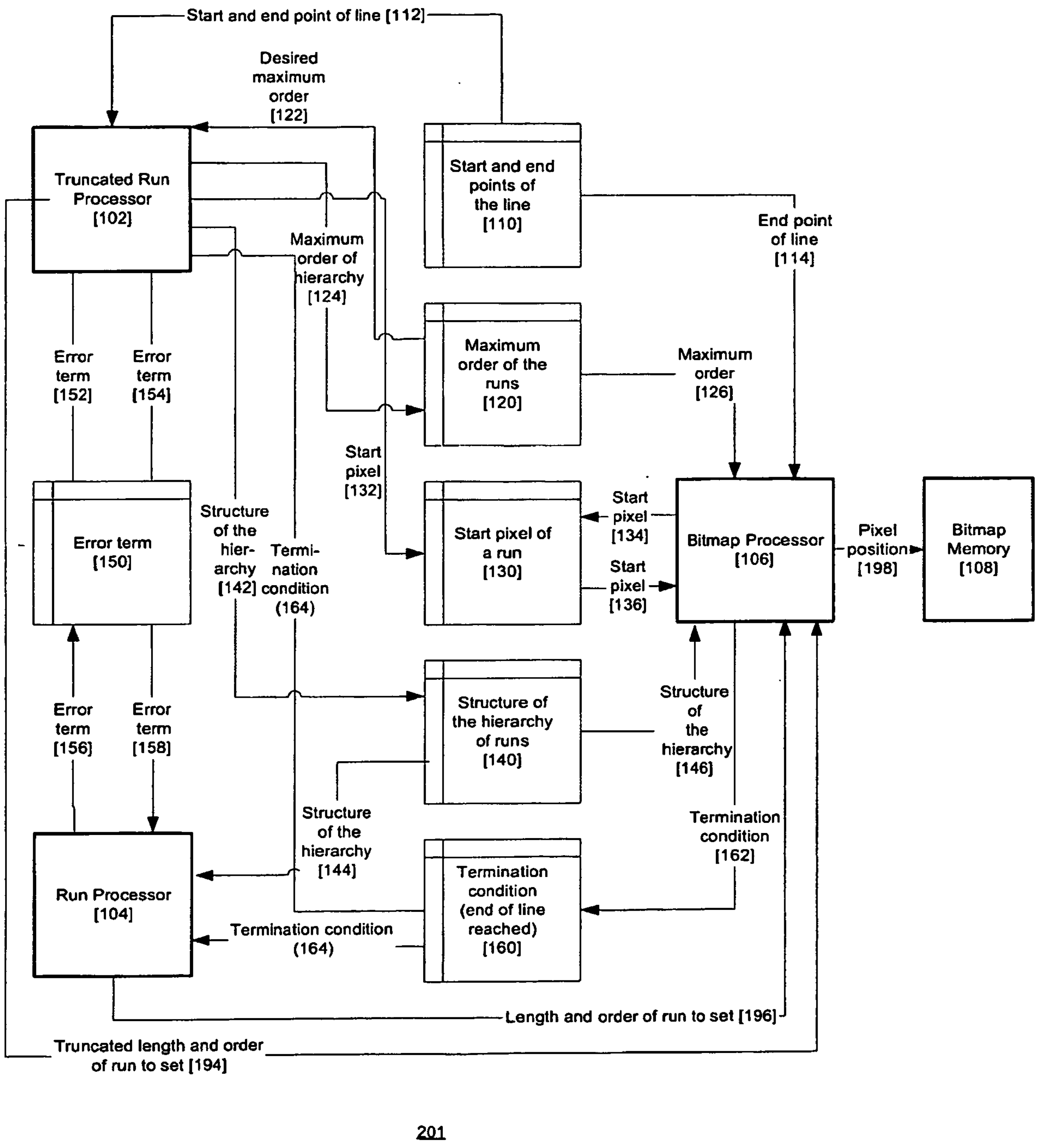

Start and end point of line [112]
Desired maximum order [122]
Truncated Run Processor [102]
Start and end points of the line [110]
End point of line [114]
Maximum order of hierarchy [124]
Maximum order of the runs [120]
Maximum order [126]
Error term [152]
Error term [154]
Start pixel [132]
Error term [150]
Structure of the hier- archy [142]
Termi- nation condition (164)
Start pixel of a run [130]
Start pixel [134]
Start pixel [136]
Bitmap Processor [106]
Pixel position [198]
Bitmap Memory [108]
Error term [156]
Error term [158]
Structure of the hierarchy of runs [140]
Structure of the hierarchy [146]
Structure of the hierarchy [144]
Termination condition [162]
Run Processor [104]
Termination condition (164)
Termination condition (end of line reached) [160]
Length and order of run to set [196]
Truncated length and order of run to set [194]
201

Fig. 2

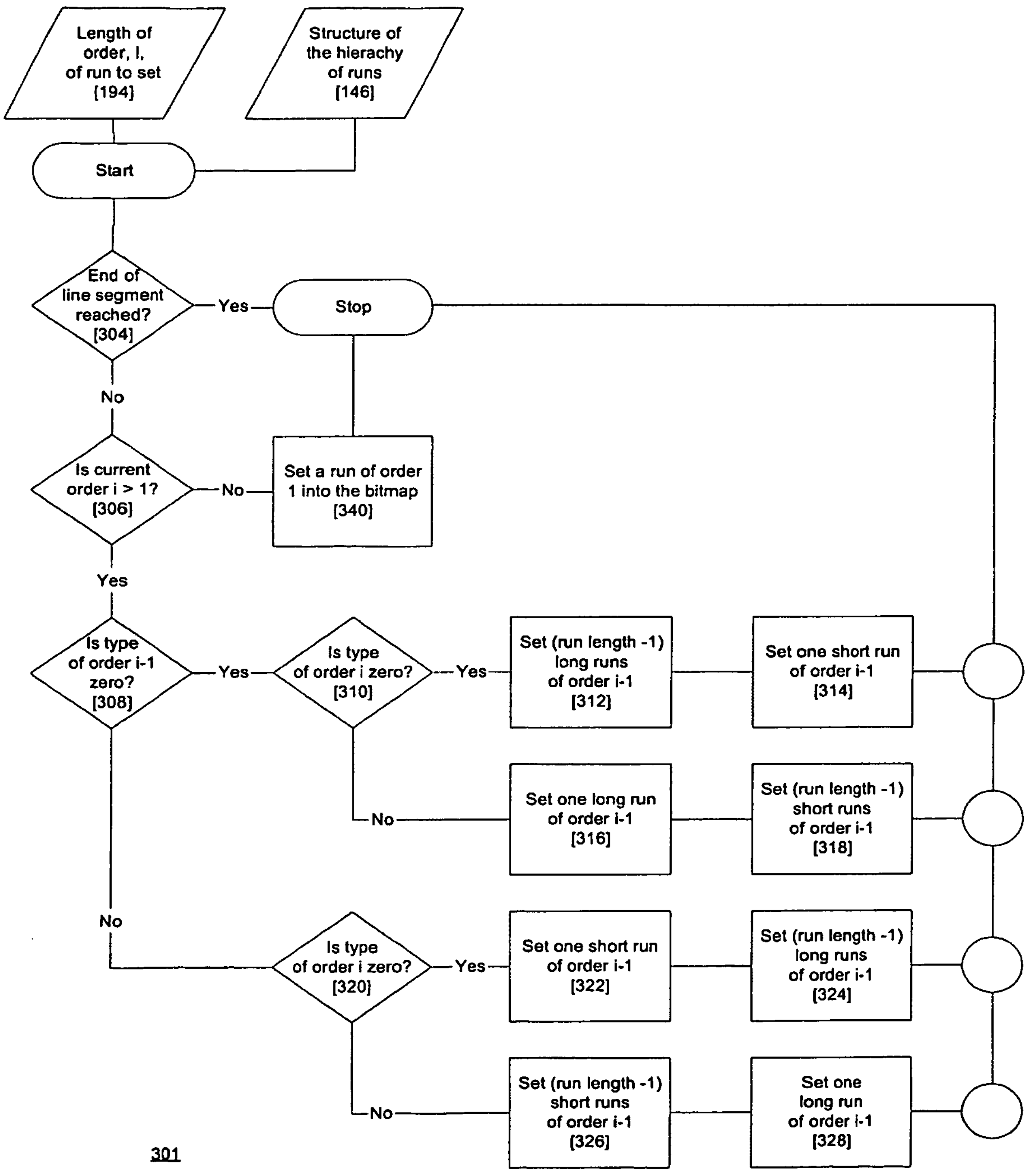

Length of order, l, of run to set [194]
Structure of the hierachy of runs [146]
Start
End of line segment reached? [304]
Yes
Stop
No
Is current order i > 1? [306]
No
Set a run of order 1 into the bitmap [340]
Yes
Is type of order i-1 zero? [308]
Yes
Is type of order i zero? [310]
Yes
Set (run length -1) long runs of order i-1 [312]
Set one short run of order i-1 [314]
No
Set one long run of order i-1 [316]
Set (run length -1) short runs of order i-1 [318]
No
Is type of order i zero? [320]
Yes
Set one short run of order i-1 [322]
Set (run length -1) long runs of order i-1 [324]
No
Set (run length -1) short runs of order i-1 [326]
Set one long run of order i-1 [328]
301

Fig. 3

10/500772

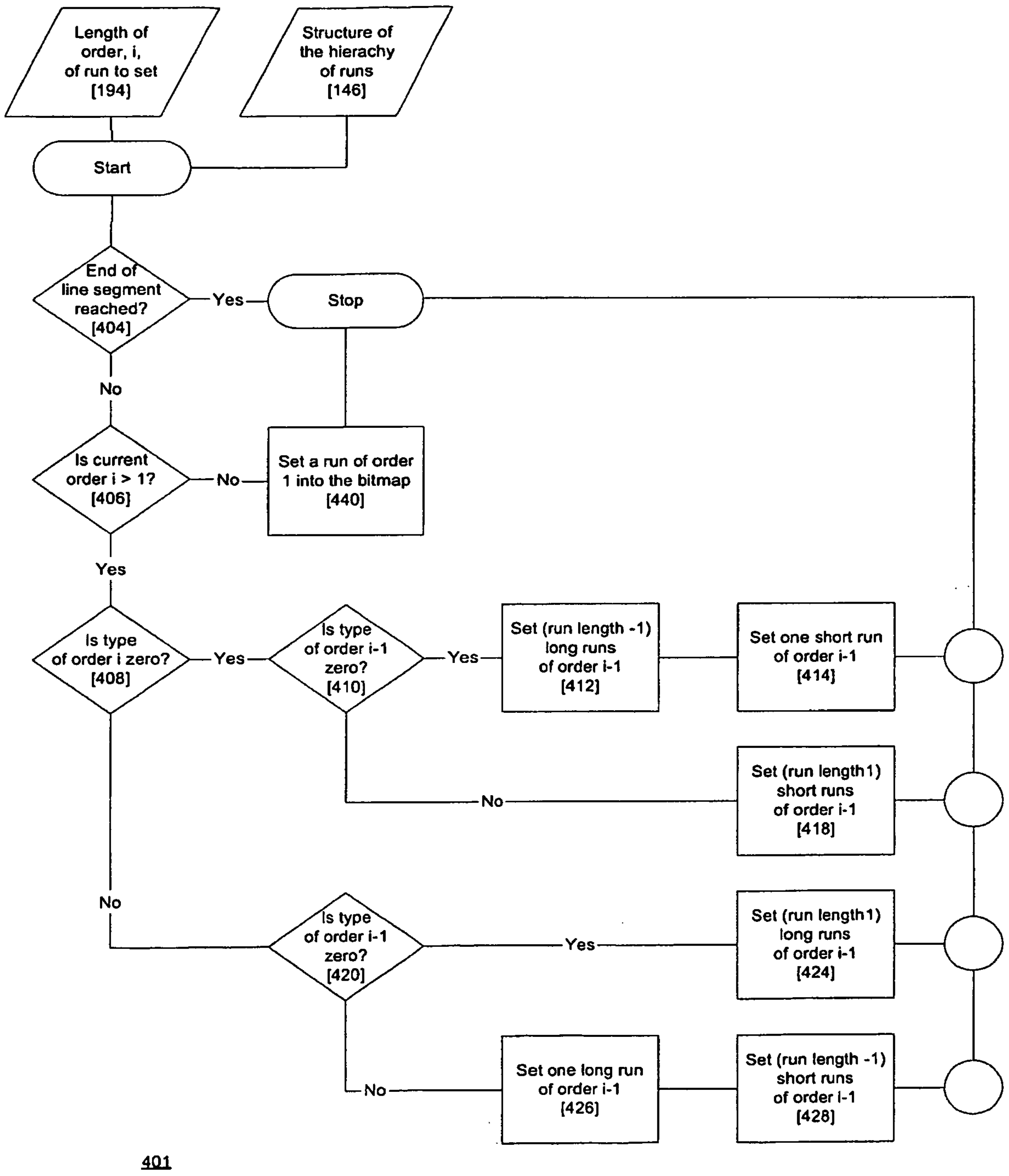

Length of order, i, of run to set [194]
Structure of the hierachy of runs [146]
Start
End of line segment reached? [404]
Yes
Stop
No
Is current order i > 1? [406]
No
Set a run of order 1 into the bitmap [440]
Yes
Is type of order i zero? [408]
Yes
Is type of order i-1 zero? [410]
Yes
Set (run length -1) long runs of order i-1 [412]
Set one short run of order i-1 [414]
No
Set (run length1) short runs of order i-1 [418]
No
Is type of order i-1 zero? [420]
Yes
Set (run length1) long runs of order i-1 [424]
No
Set one long run of order i-1 [426]
Set (run length -1) short runs of order i-1 [428]
401

Fig. 4

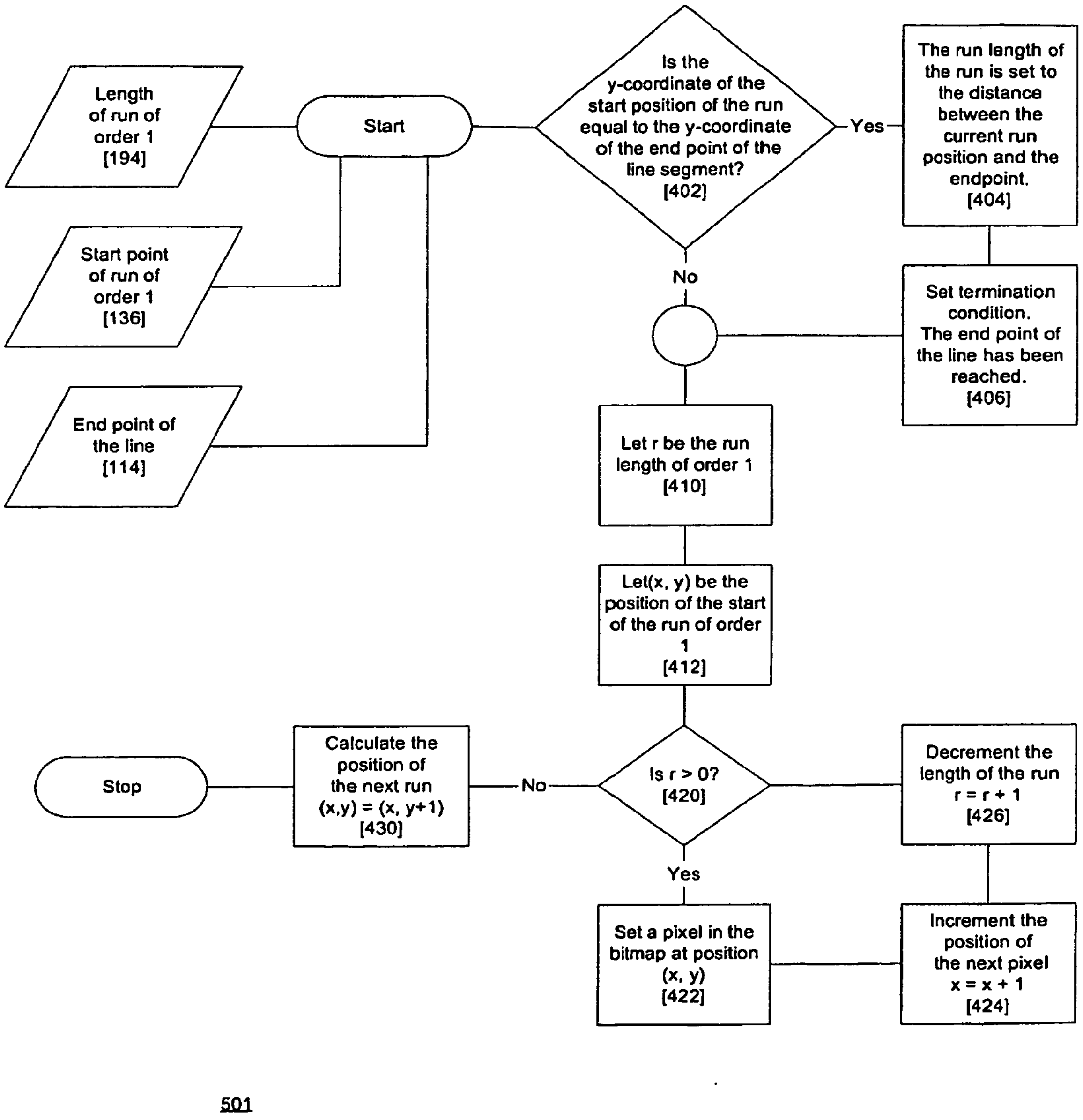

Length of run of order 1 [194]
Start point of run of order 1 [136]
End point of the line [114]
Start
Is the y-coordinate of the start position of the run equal to the y-coordinate of the end point of the line segment? [402]
Yes
The run length of the run is set to the distance between the current run position and the endpoint. [404]
Set termination condition. The end point of the line has been reached. [406]
No
Let r be the run length of order 1 [410]
Let(x, y) be the position of the start of the run of order 1 [412]
Is r > 0? [420]
No
Calculate the position of the next run (x,y) = (x, y+1) [430]
Stop
Yes
Set a pixel in the bitmap at position (x, y) [422]
Increment the position of the next pixel x = x + 1 [424]
Decrement the length of the run r = r + 1 [426]
501

Fig. 5

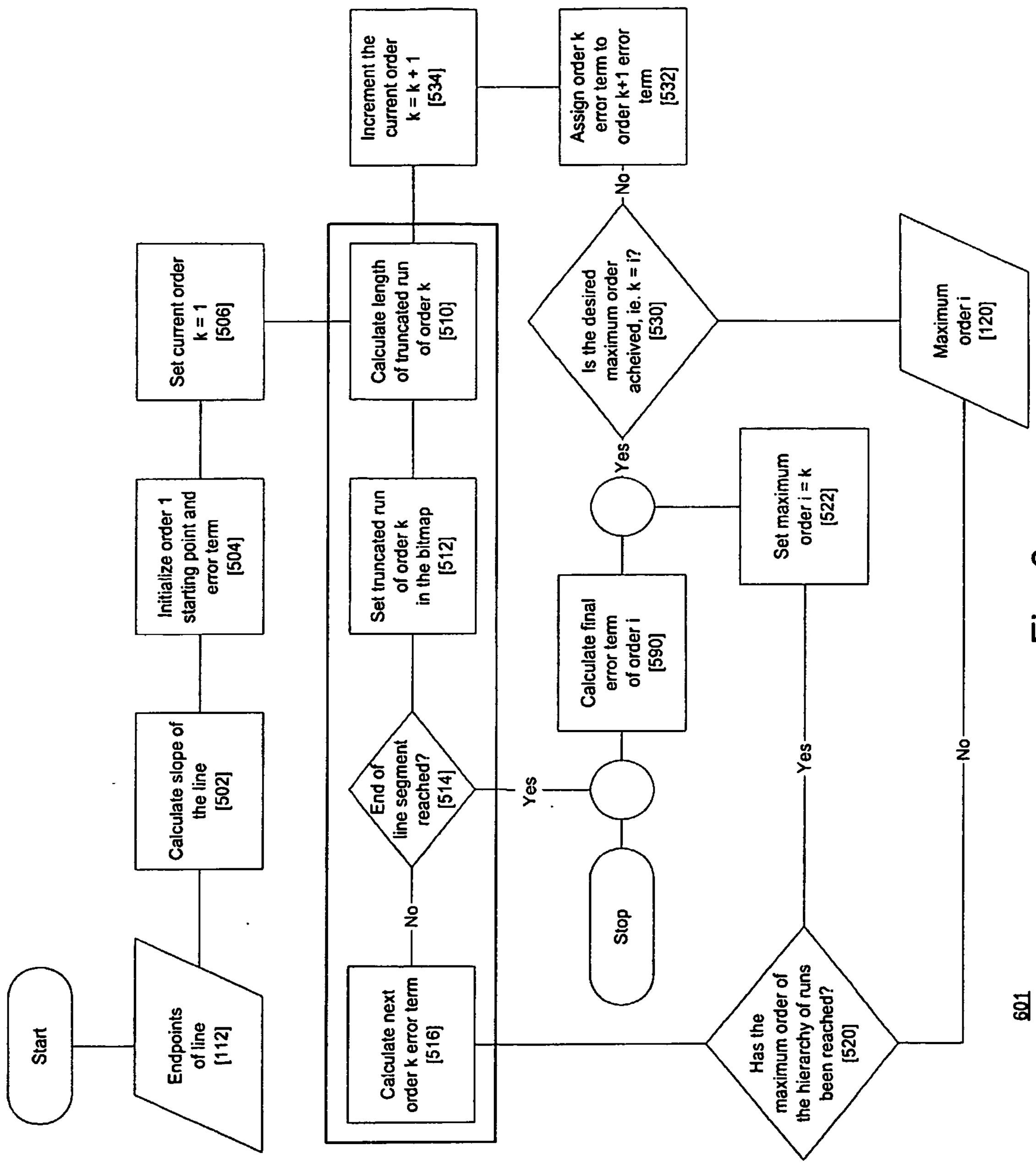

Start
Endpoints of line [112]
Calculate slope of the line [502]
Initialize order 1 starting point and error term [504]
Set current order k = 1 [506]
Calculate length of truncated run of order k [510]
Set truncated run of order k in the bitmap [512]
End of line segment reached? [514]
No
Yes
Calculate next order k error term [516]
Stop
Calculate final error term of order i [590]
Yes
Is the desired maximum order acheived, ie. k = i? [530]
No
Assign order k error term to order k+1 error term [532]
Increment the current order k = k + 1 [534]
Set maximum order i = k [522]
Has the maximum order of the hierarchy of runs been reached? [520]
Yes
No
Maximum order i [120]
601

Fig. 6

10/500772

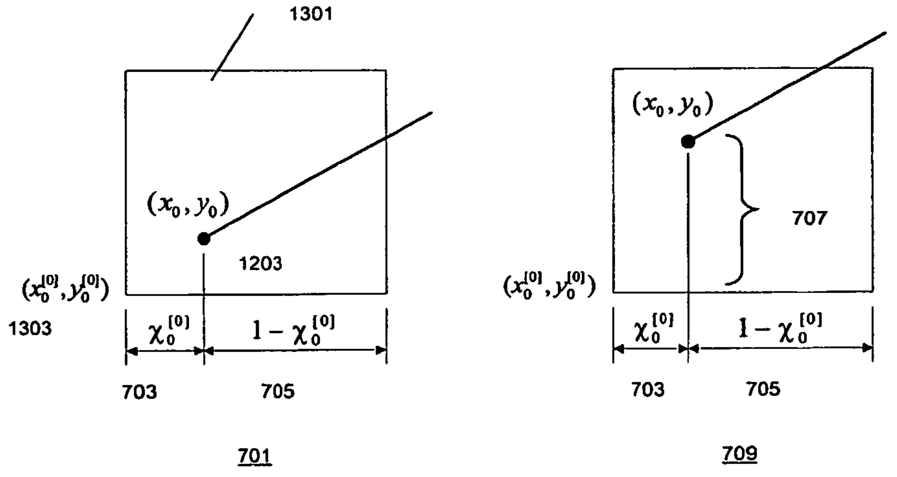


Fig. 7

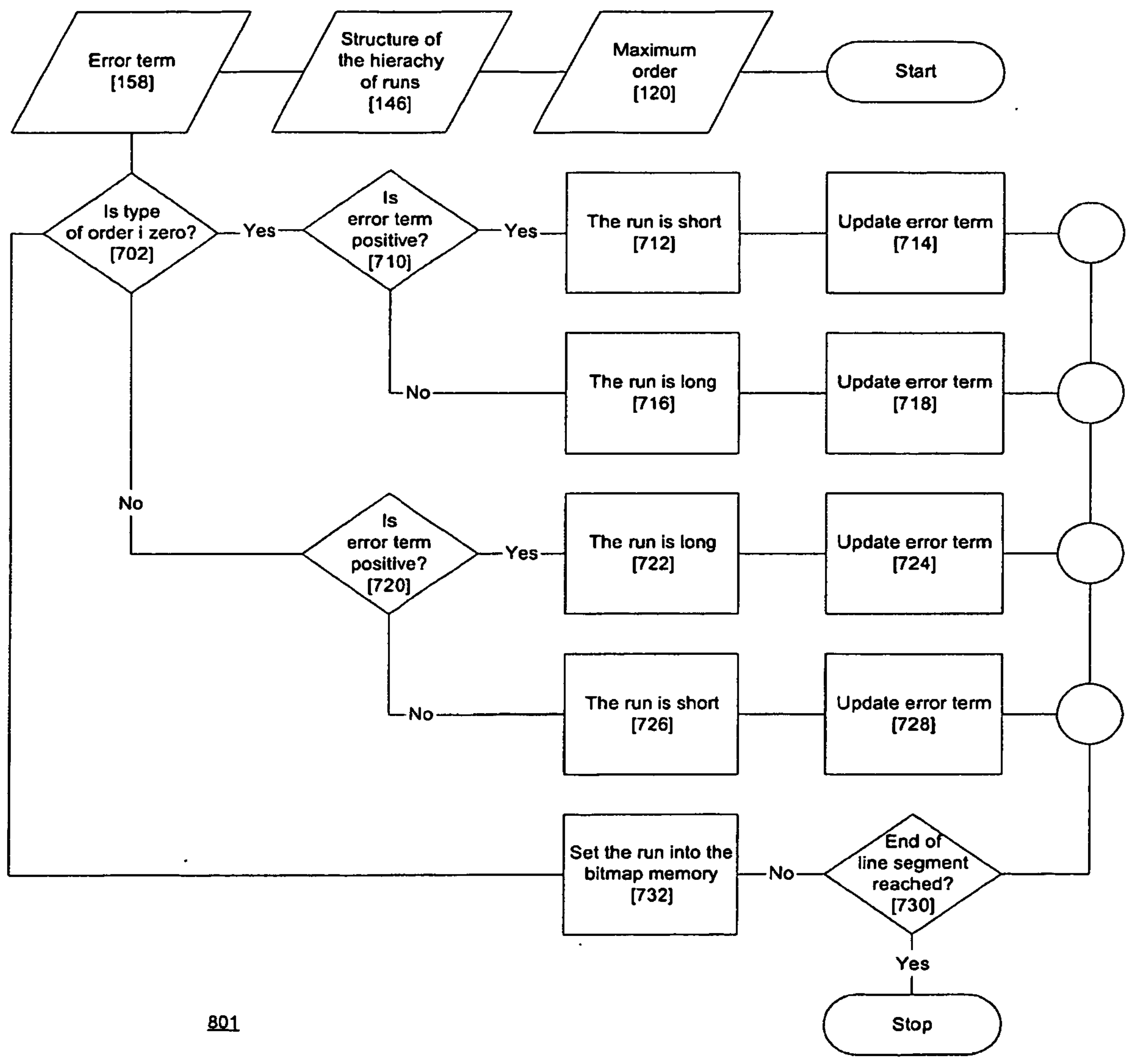

Error term [158]
Structure of the hierachy of runs [146]
Maximum order [120]
Start
Is type of order i zero? [702]
Yes
Is error term positive? [710]
Yes
The run is short [712]
Update error term [714]
No
The run is long [716]
Update error term [718]
No
Is error term positive? [720]
Yes
The run is long [722]
Update error term [724]
No
The run is short [726]
Update error term [728]
Set the run into the bitmap memory [732]
No
End of line segment reached? [730]
Yes
Stop
801

Fig. 8

10/500772

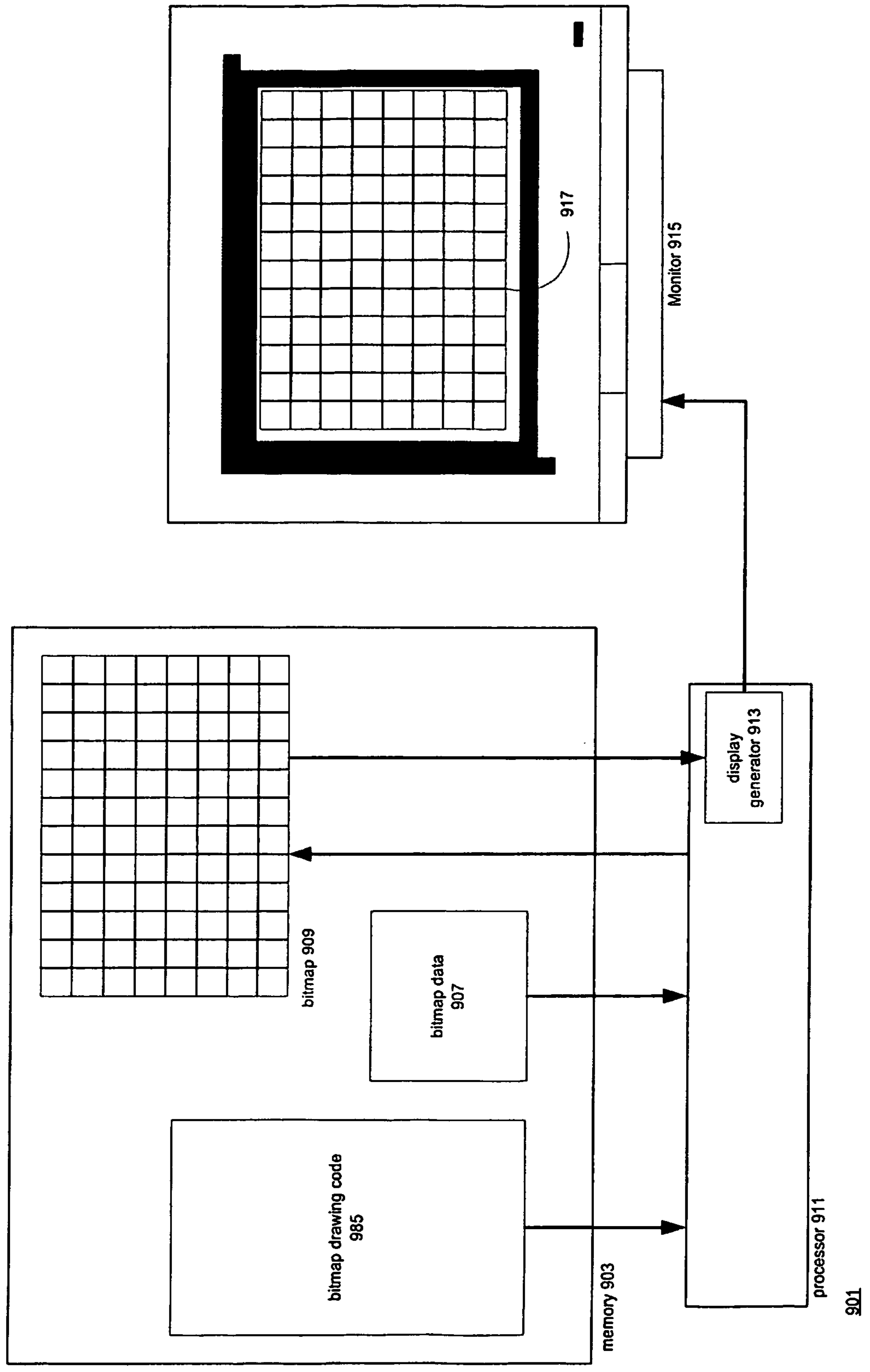


Fig. 9

10/500772

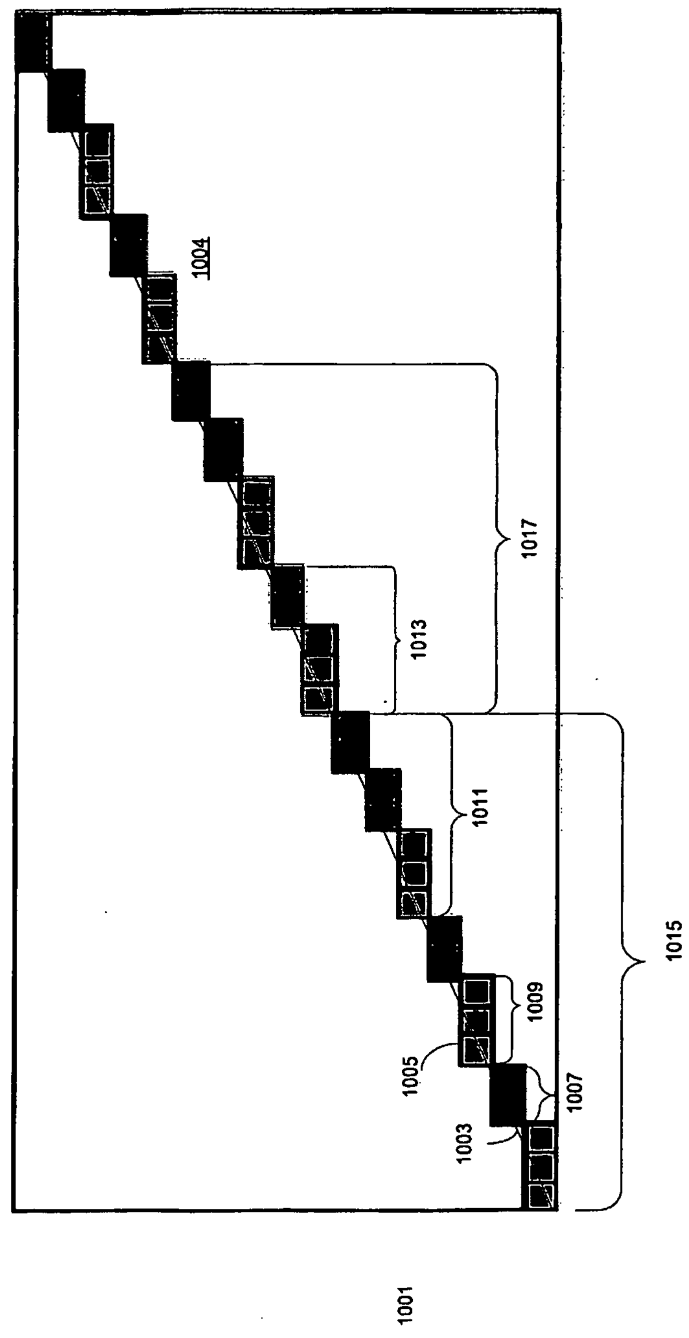
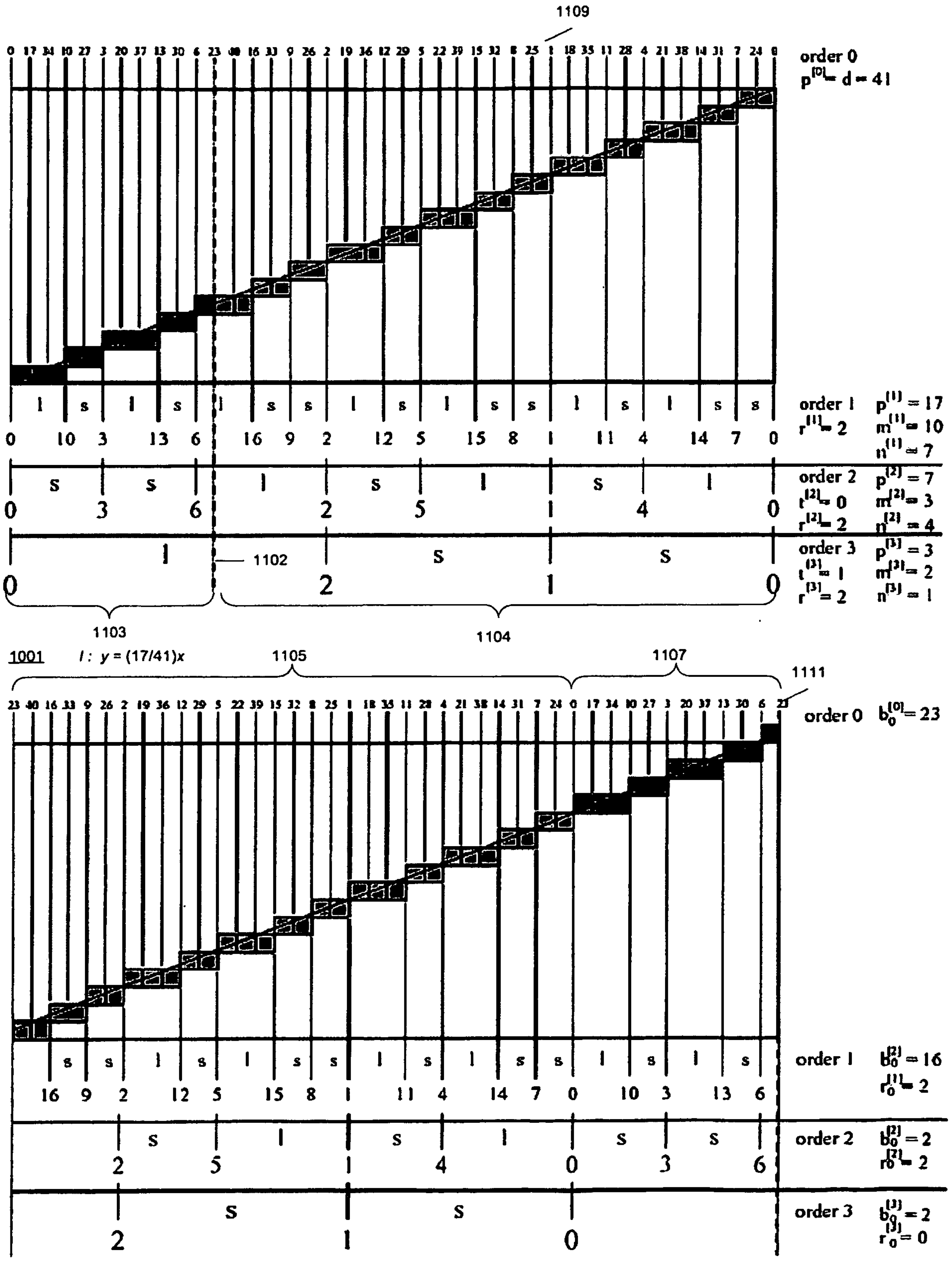


Fig. 10



1109
order 0
$p^{[0]} = d = 41$
order 1
$p^{[1]} = 17$
$r^{[1]} = 2$
$m^{[1]} = 10$
$n^{[1]} = 7$
order 2
$p^{[2]} = 7$
$t^{[2]} = 0$
$m^{[2]} = 3$
$r^{[2]} = 2$
$n^{[2]} = 4$
order 3
$p^{[3]} = 3$
$t^{[3]} = 1$
$m^{[3]} = 2$
$r^{[3]} = 2$
$n^{[3]} = 1$
1102
1103
1104
1001
$l: y = (17/41)x$
1105
1107
1111
order 0
$b_0^{[0]} = 23$
order 1
$b_0^{[1]} = 16$
$r_0^{[1]} = 2$
order 2
$b_0^{[2]} = 2$
$r_0^{[2]} = 2$
order 3
$b_0^{[3]} = 2$
$r_0^{[3]} = 0$

Fig. 11

10/500772

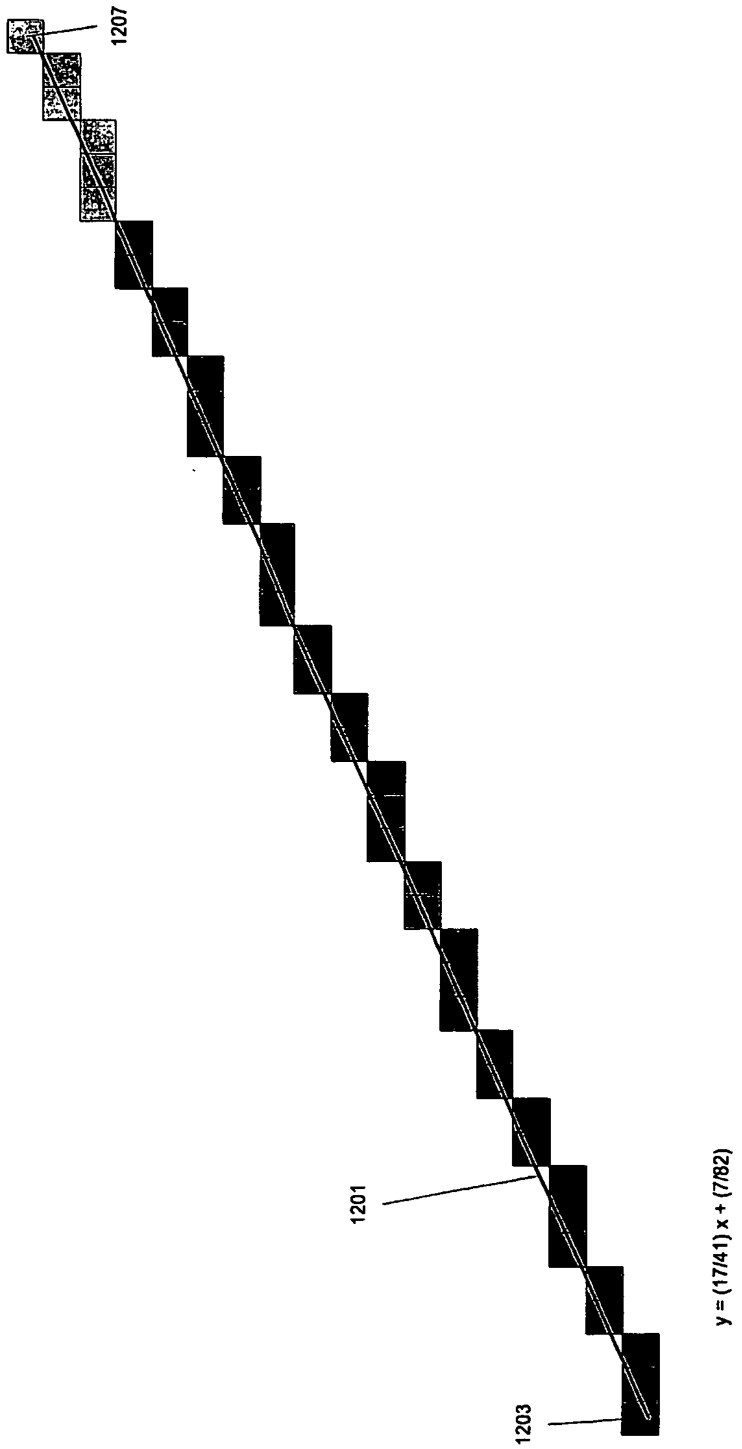


Fig. 12

10/500772

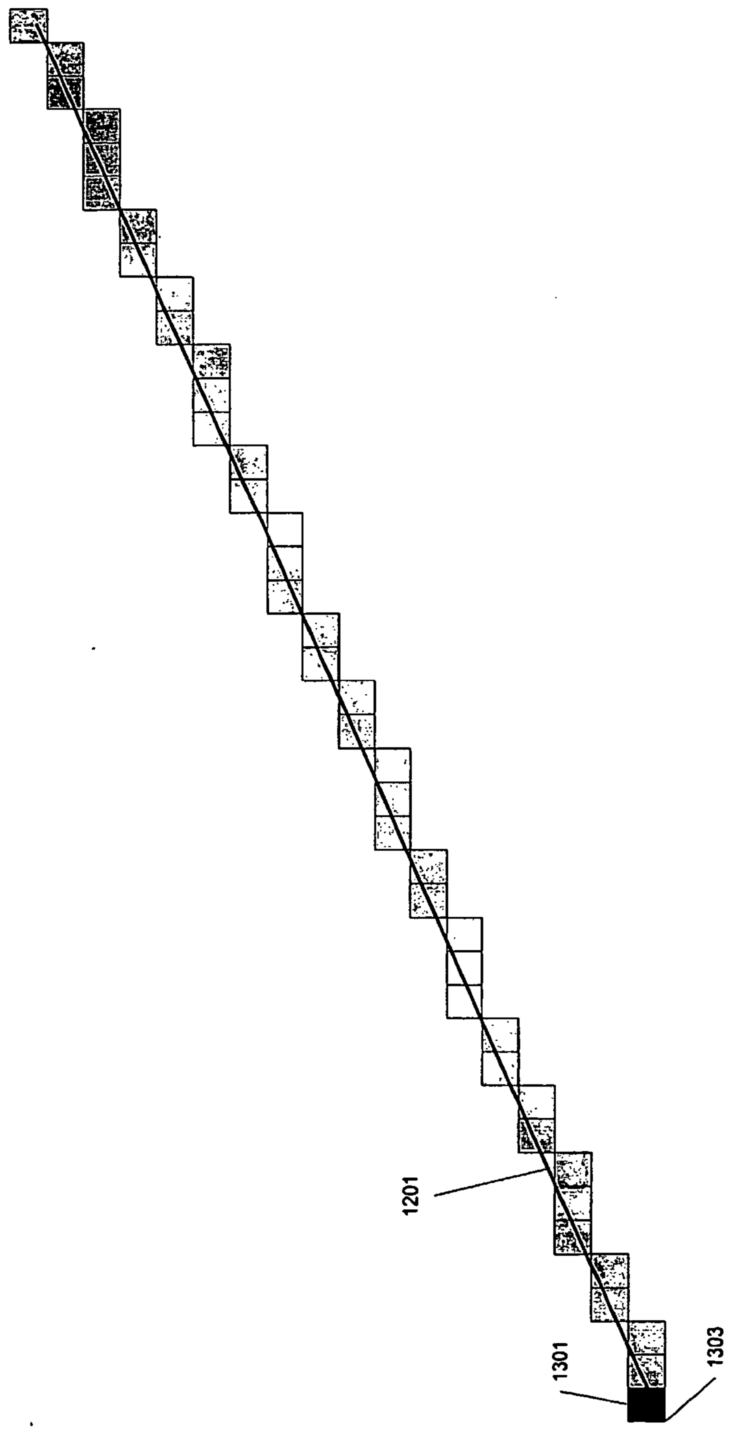


Fig. 13

10/500772

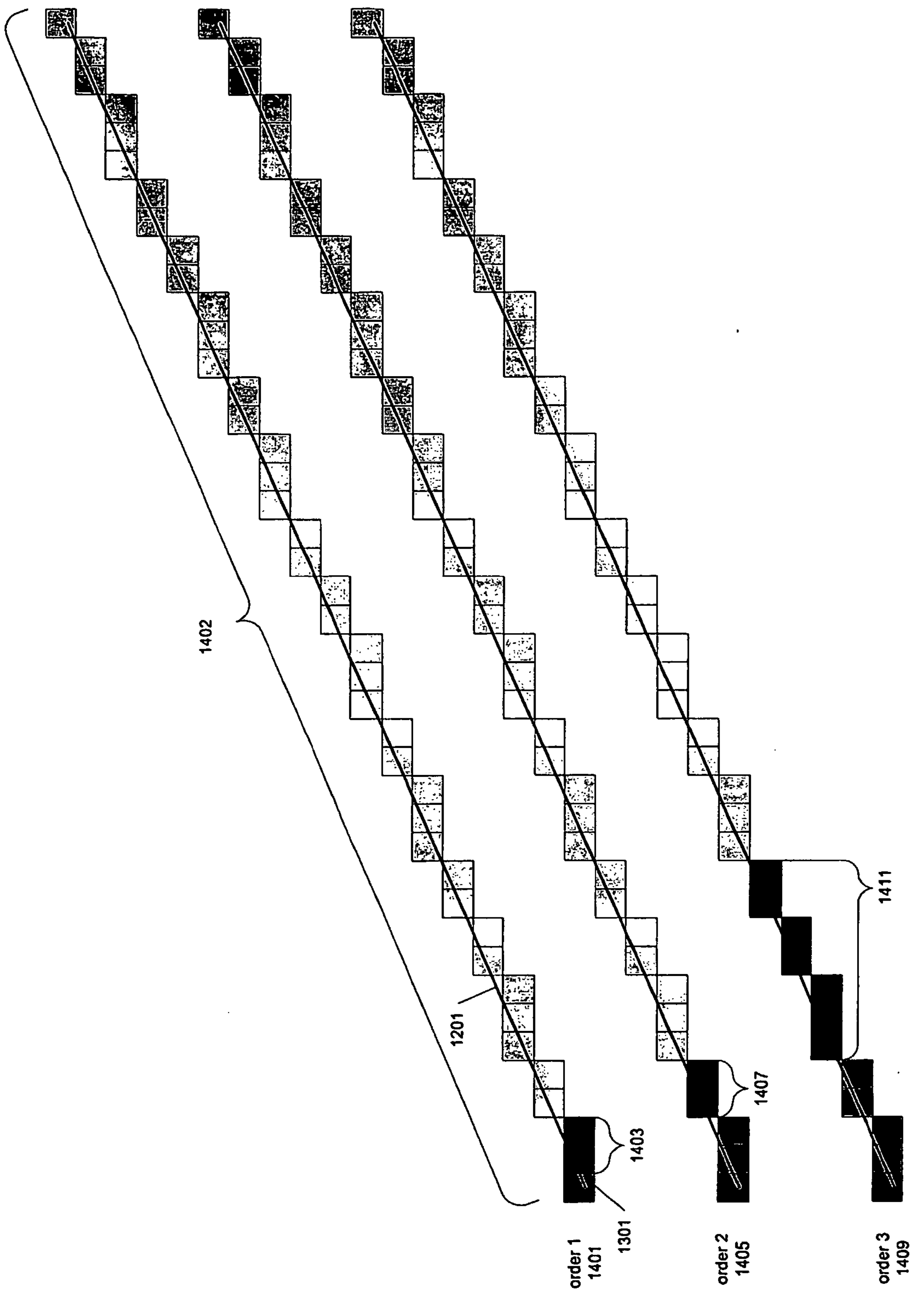


Fig. 14

10/500772

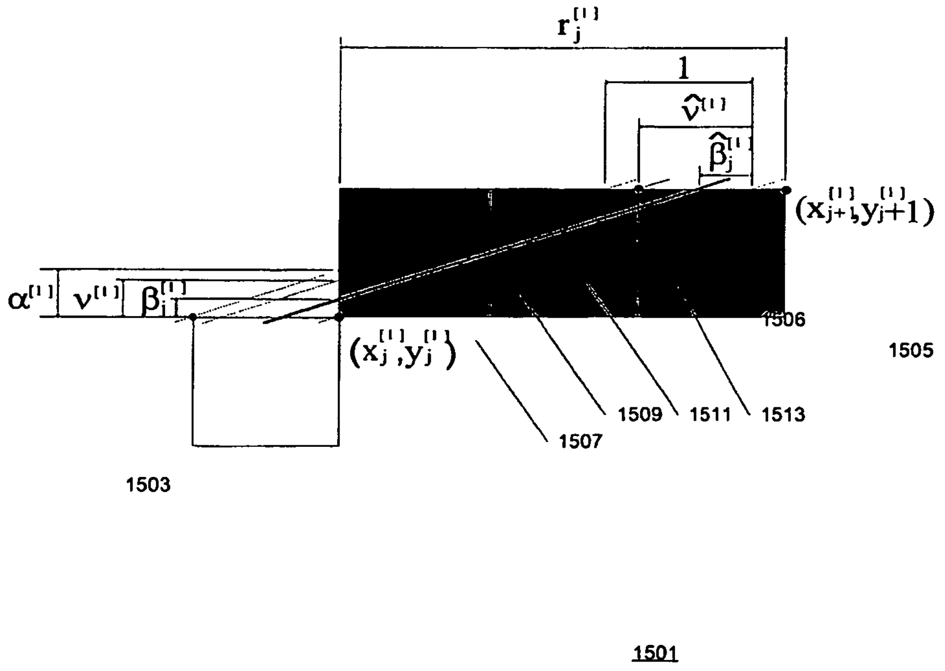
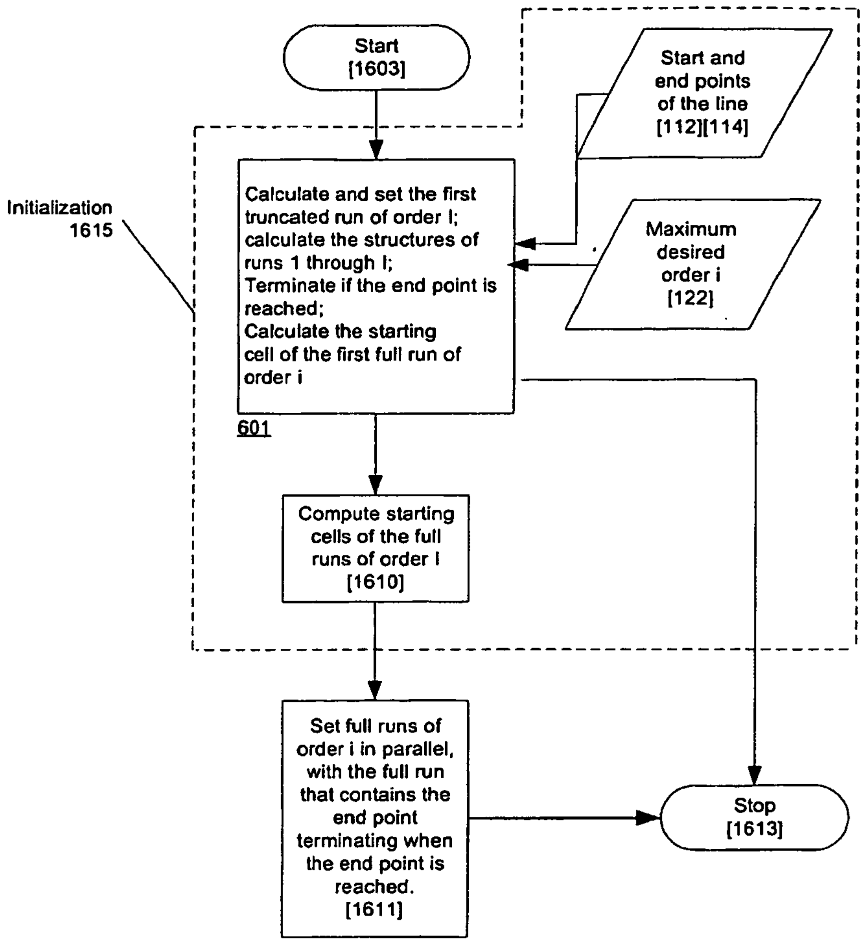


Fig. 15


Start
[1603]
Start and
end points
of the line
[112][114]
Initialization
1615
Calculate and set the first truncated run of order I; calculate the structures of runs 1 through I; Terminate if the end point is reached; Calculate the starting cell of the first full run of order i
Maximum
desired
order i
[122]
601
Compute starting cells of the full runs of order I
[1610]
Set full runs of order i in parallel, with the full run that contains the end point terminating when the end point is reached.
[1611]
Stop
[1613]

1601

Fig. 16

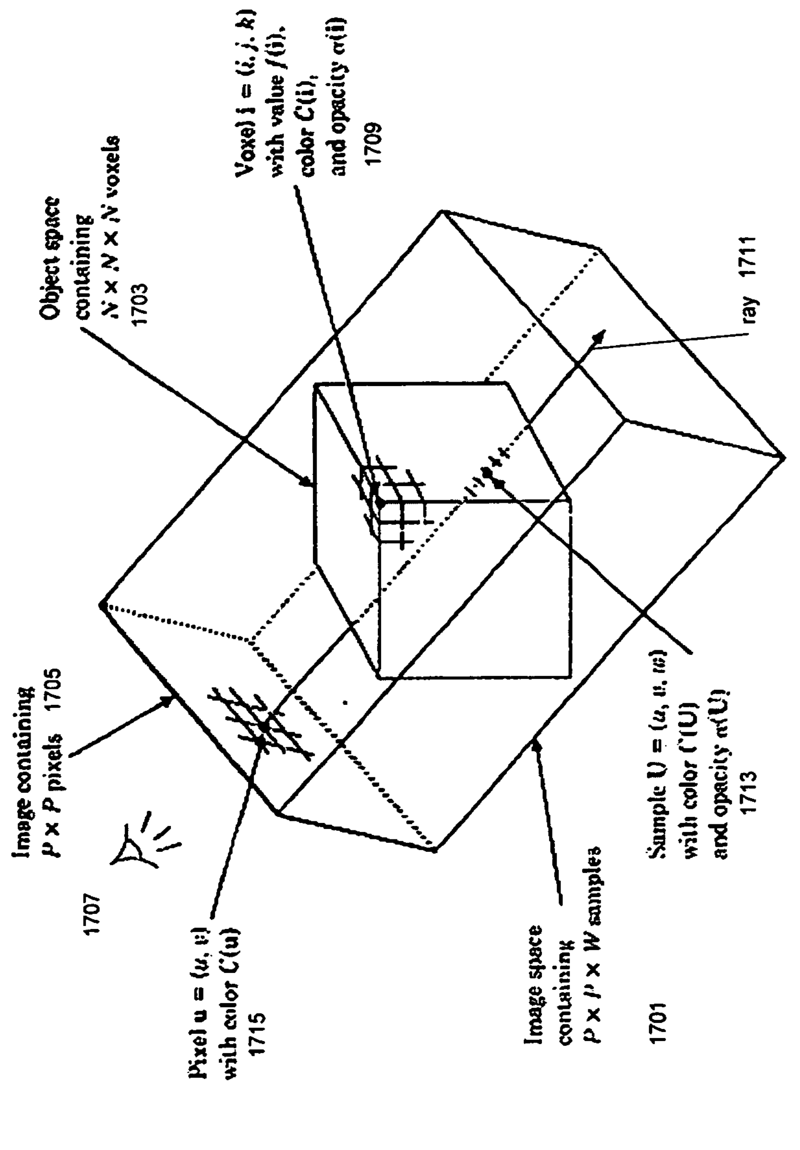

Object space containing $N \times N \times N$ voxels 1703
Voxel $\mathbf{i} = (i, j, k)$ with value $f(\mathbf{i})$, color $C(\mathbf{i})$, and opacity $\alpha(\mathbf{i})$ 1709
ray 1711
Image containing $P \times P$ pixels 1705
1707
Pixel $\mathbf{u} = (u, v)$ with color $C(\mathbf{u})$ 1715
Image space containing $P \times P \times W$ samples
1701
Sample $\mathbf{U} = (u, v, w)$ with color $C(\mathbf{U})$ and opacity $\alpha(\mathbf{U})$ 1713

Fig. 17

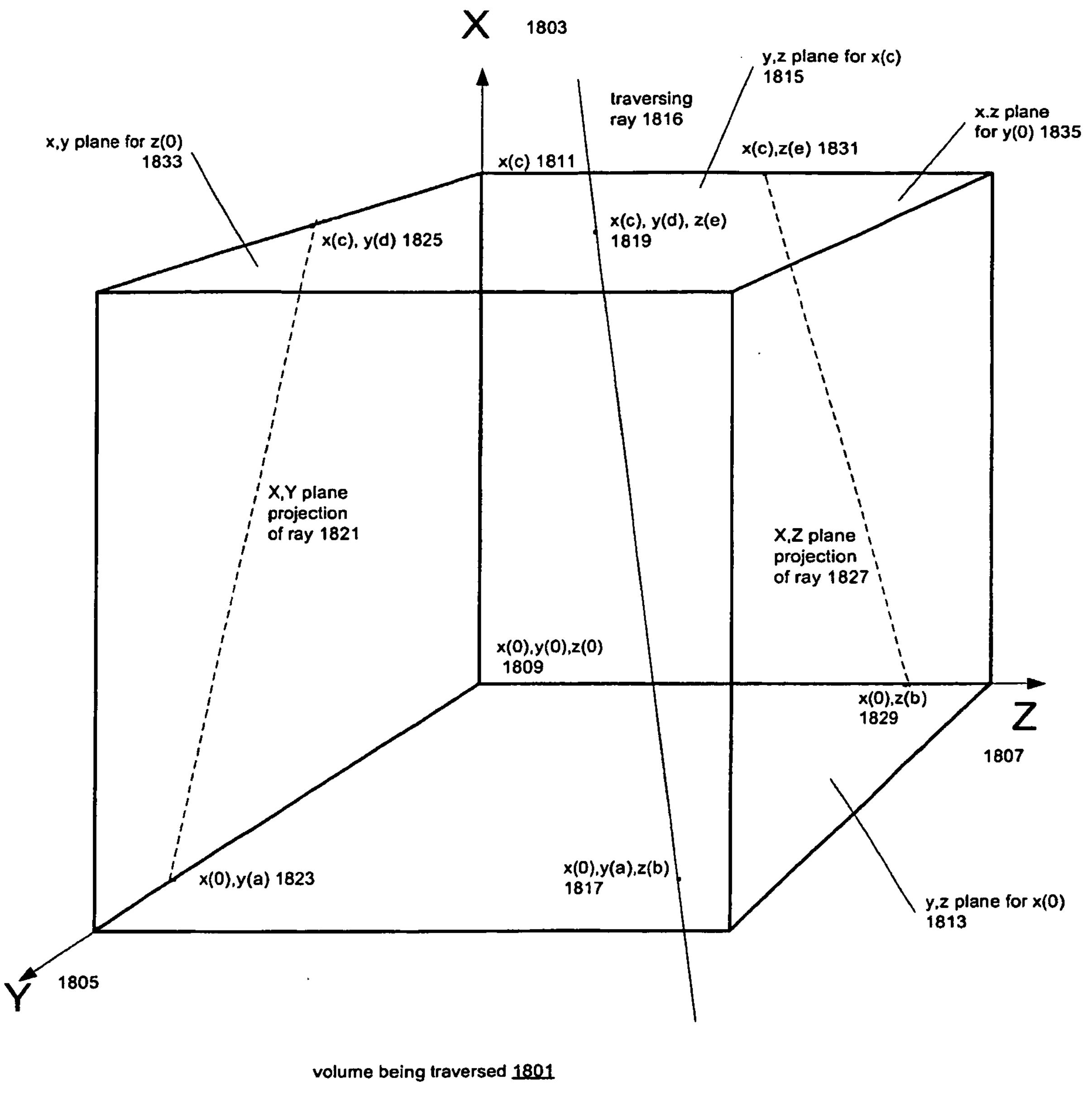

X 1803
y,z plane for x(c) 1815
traversing ray 1816
x.z plane for y(0) 1835
x,y plane for z(0) 1833
x(c) 1811
x(c),z(e) 1831
x(c), y(d) 1825
x(c), y(d), z(e) 1819
X,Y plane projection of ray 1821
X,Z plane projection of ray 1827
x(0),y(0),z(0) 1809
x(0),z(b) 1829
Z 1807
x(0),y(a) 1823
x(0),y(a),z(b) 1817
y,z plane for x(0) 1813
Y 1805
volume being traversed 1801

Fig. 18

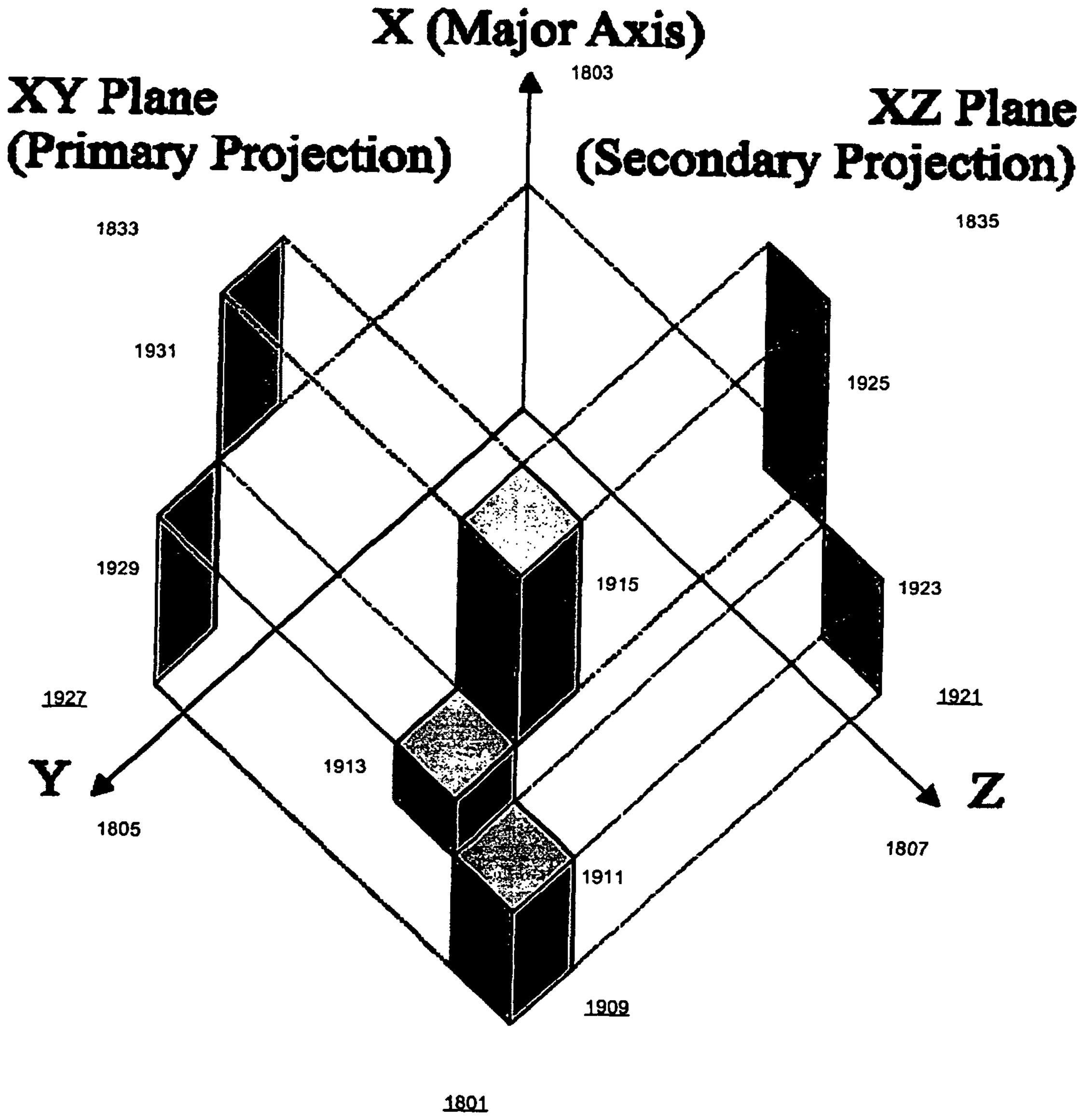

X (Major Axis)
1803
XY Plane
(Primary Projection)
XZ Plane
(Secondary Projection)
1833
1835
1931
1925
1929
1915
1923
1927
1921
1913
Y
Z
1805
1807
1911
1909
1801

Fig. 19

```
$(x, y, z) = ray.startPoint$

Get the first run length in each projected ray path.
$r_{XY} = projection_{XY}.firstRunLength()$
$r_{XZ} = projection_{XZ}.firstRunLength()$

while unterminated
    if $r_{XY} < r_{XZ}$
        $subdivision.traverseRun(r_{XY}, x, y, z)$

        Calculate the position of the next run.
        $x+ = r_{XY}$
        $y++$

        Shorten the corresponding XZ run.
        $r_{XZ}- = r_{XY}$

        Get the next XY run length.
        $r_{XY} = projection_{XY}.nextRunLength()$

    else if $r_{XY} > r_{XZ}$
        $subdivision.traverseRun(r_{XZ}, x, y, z)$

        Calculate the position of the next run.
        $x+ = r_{XZ}$
        $z++$

        Shorten the corresponding XY run.
        $r_{XY}- = r_{XZ}$

        Get the next XZ run length.
        $r_{XZ} = projection_{XZ}.nextRunLength()$

    elseThe XY and XZ runs have the same length.
        $subdivision.traverseRun(r_{XZ}, x, y, z)$

        Calculate position of next run.
        $x+ = r_{XZ}$
        $y++$
        $z++$

        Get the next XY and XZ run length.
        $r_{XY} = projection_{XY}.nextRunLength()$
        $r_{XZ} = projection_{XZ}.nextRunLength()$
```

2001

Fig. 20

10/500772

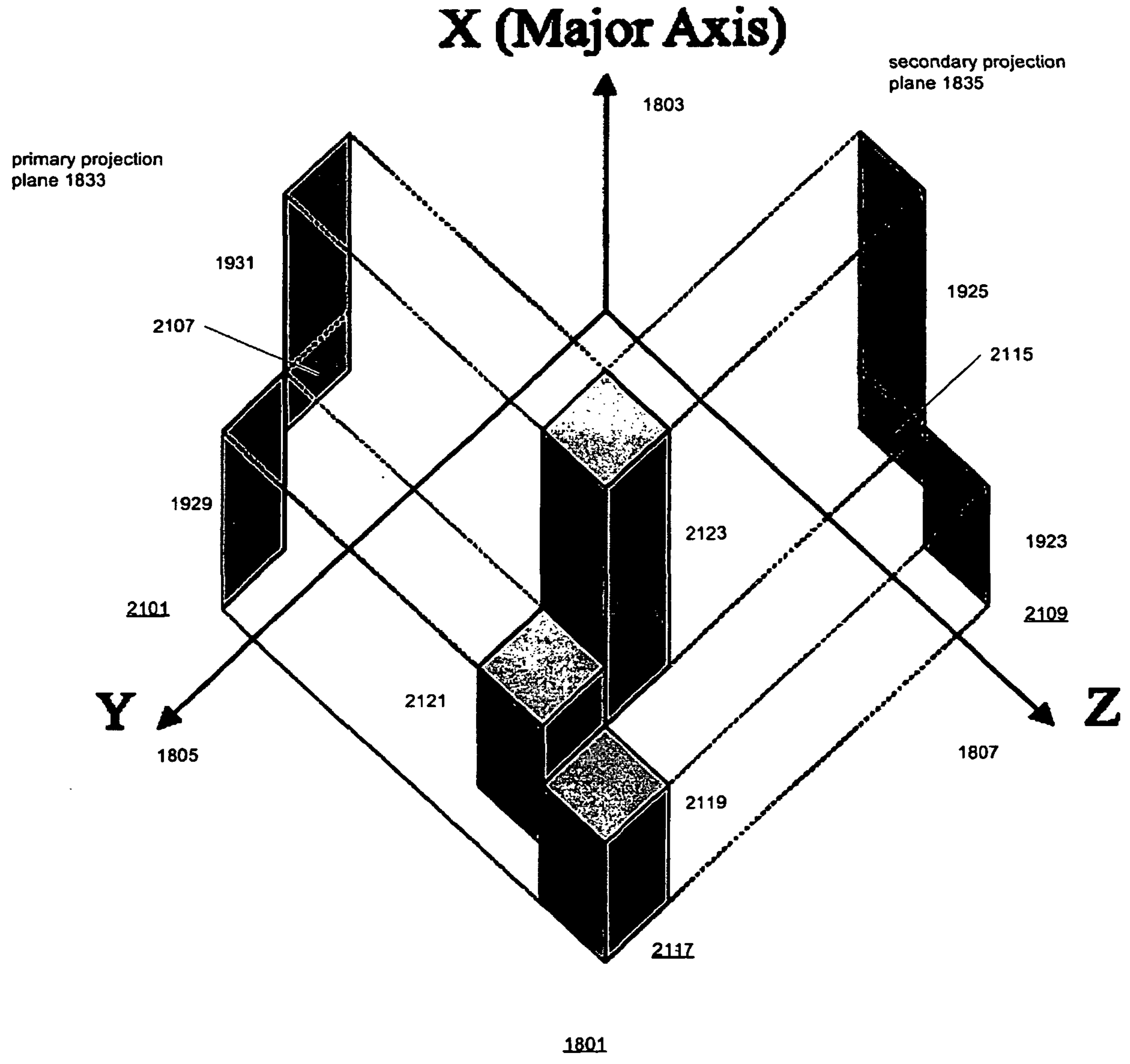
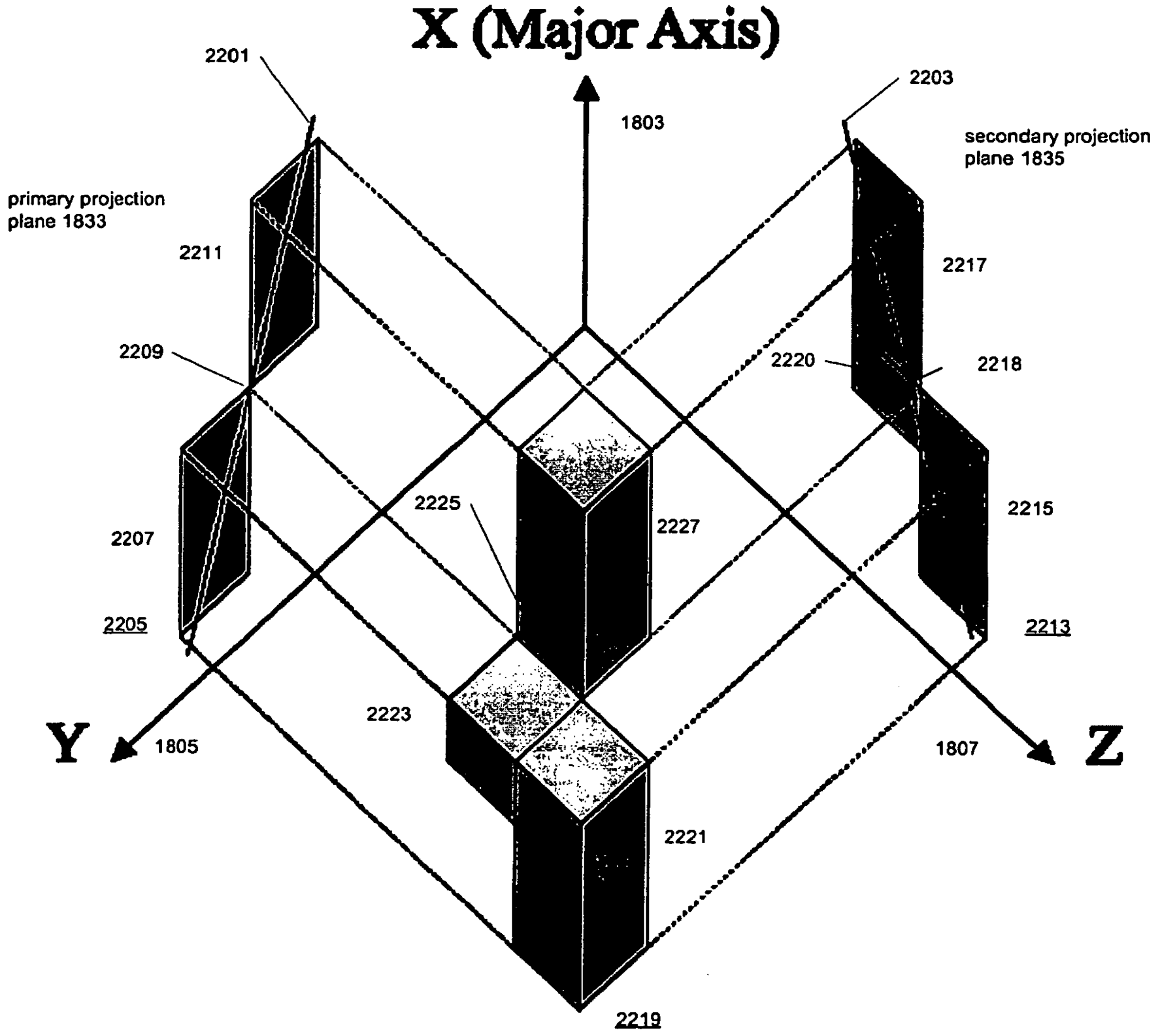


Fig. 21


X (Major Axis)
2201
2203
1803
secondary projection plane 1835
primary projection plane 1833
2211
2217
2209
2220
2218
2225
2227
2215
2207
2205
2213
2223
Y
1805
1807
Z
2221
2219

1801

Fig. 22

```
if $r_{XY} < r_{XZ}$
    if $projection_{XY}.\beta$ is non–zero
        No edge intersection.
        $subdivision.traverseRun$
                ($r_{XY} + 1, x - 1, y, z$)
    else
        Edge intersection.
        $subdivision.traverseRun(r_{XY}, x, y, z)$

    $x+ = r_{XY}$
    $y + +$
    $r_{XZ} - = r_{XY}$
    $r_{XY} = projection_{XY}.nextRunLength()$
```

2301

Fig. 23

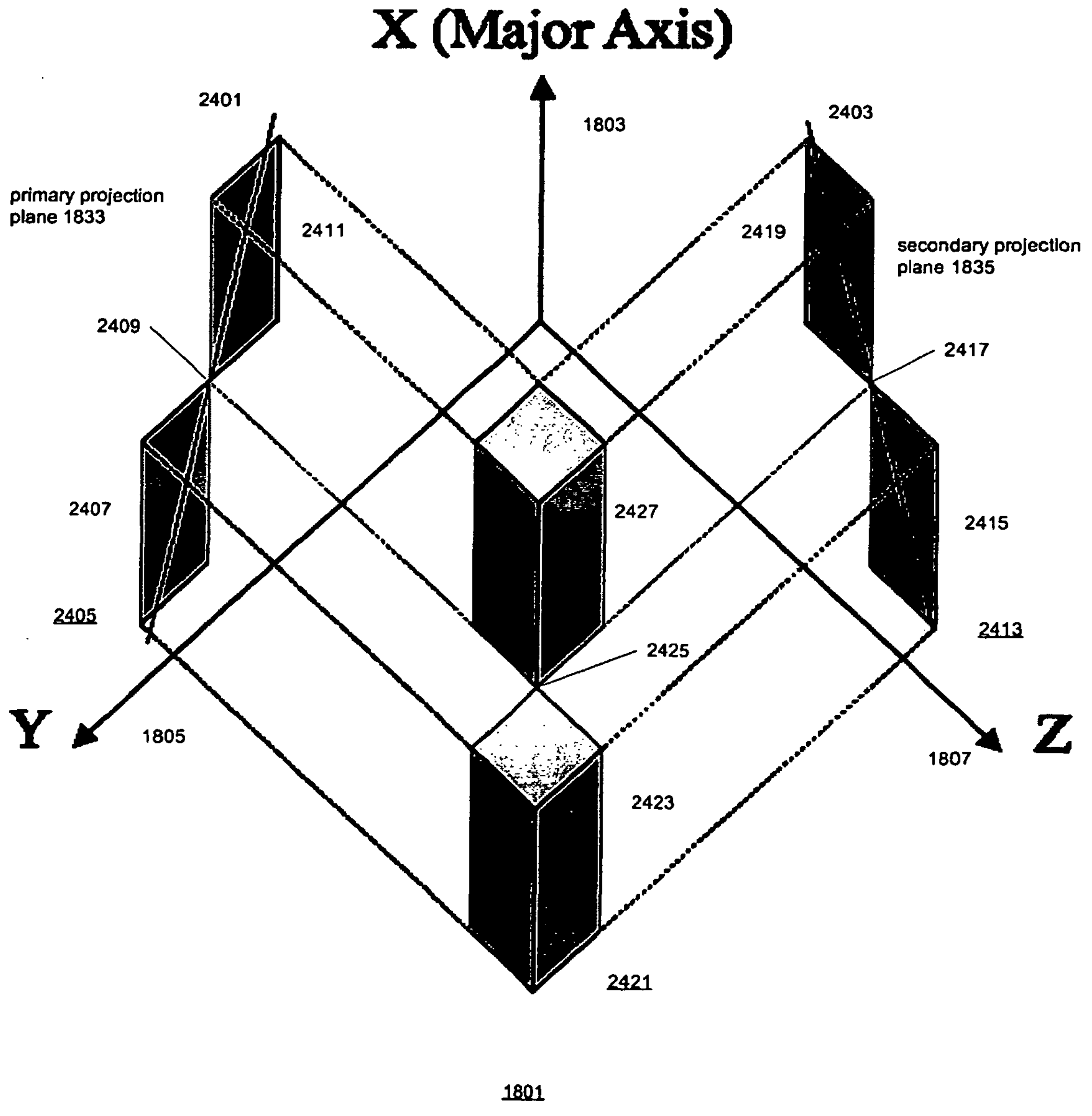

X (Major Axis)
2401
1803
2403
primary projection plane 1833
2411
2419
secondary projection plane 1835
2409
2417
2407
2427
2415
2405
2413
2425
Y
1805
1807
Z
2423
2421
1801

Fig. 24

```
if $r_{XY} == r_{XZ}$
    if $projection_{XZ}.\hat{\beta} < projection_{XY}.\hat{\beta}$
        $subdivision.traverseCell(x, y, z - 1)$
    else if $projection_{XY}.\hat{\beta} < projection_{XZ}.\hat{\beta}$
        $subdivision.traverseCell(x, y - 1, z)$
    else $projection_{XY}.\hat{\beta} == projection_{XZ}.\hat{\beta}$
        if $projection_{XY}.\hat{\beta}$ is zero
            No corner intersection.
            $subdivision.traverseRun$
                            $(r_{XY} + 1, x - 1, y, z)$
        else
            Corner intersection.
            $subdivision.traverseRun(r_{XY}, x, y, z)$

    $x+ = r_{XY}$
    $r_{XY} = projection_{XY}.nextRunLength()$
    $r_{XZ} = projection_{XZ}.nextRunLength()$
```

2501

Fig. 25

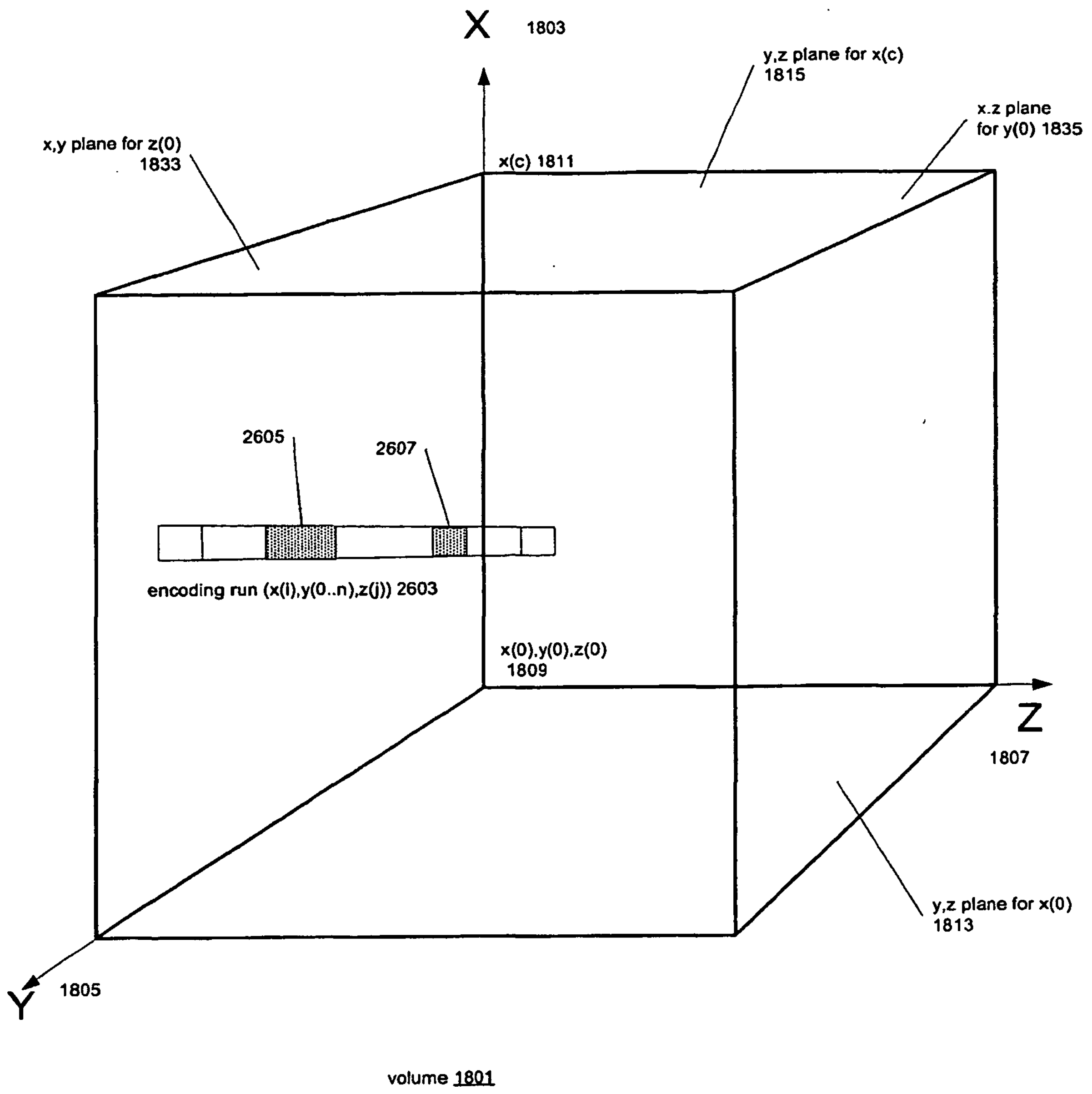

X 1803
y,z plane for x(c) 1815
x.z plane for y(0) 1835
x,y plane for z(0) 1833
x(c) 1811
2605
2607
encoding run (x(i),y(0..n),z(j)) 2603
x(0),y(0),z(0) 1809
Z 1807
y,z plane for x(0) 1813
Y 1805
volume 1801

Fig. 26

10/500772

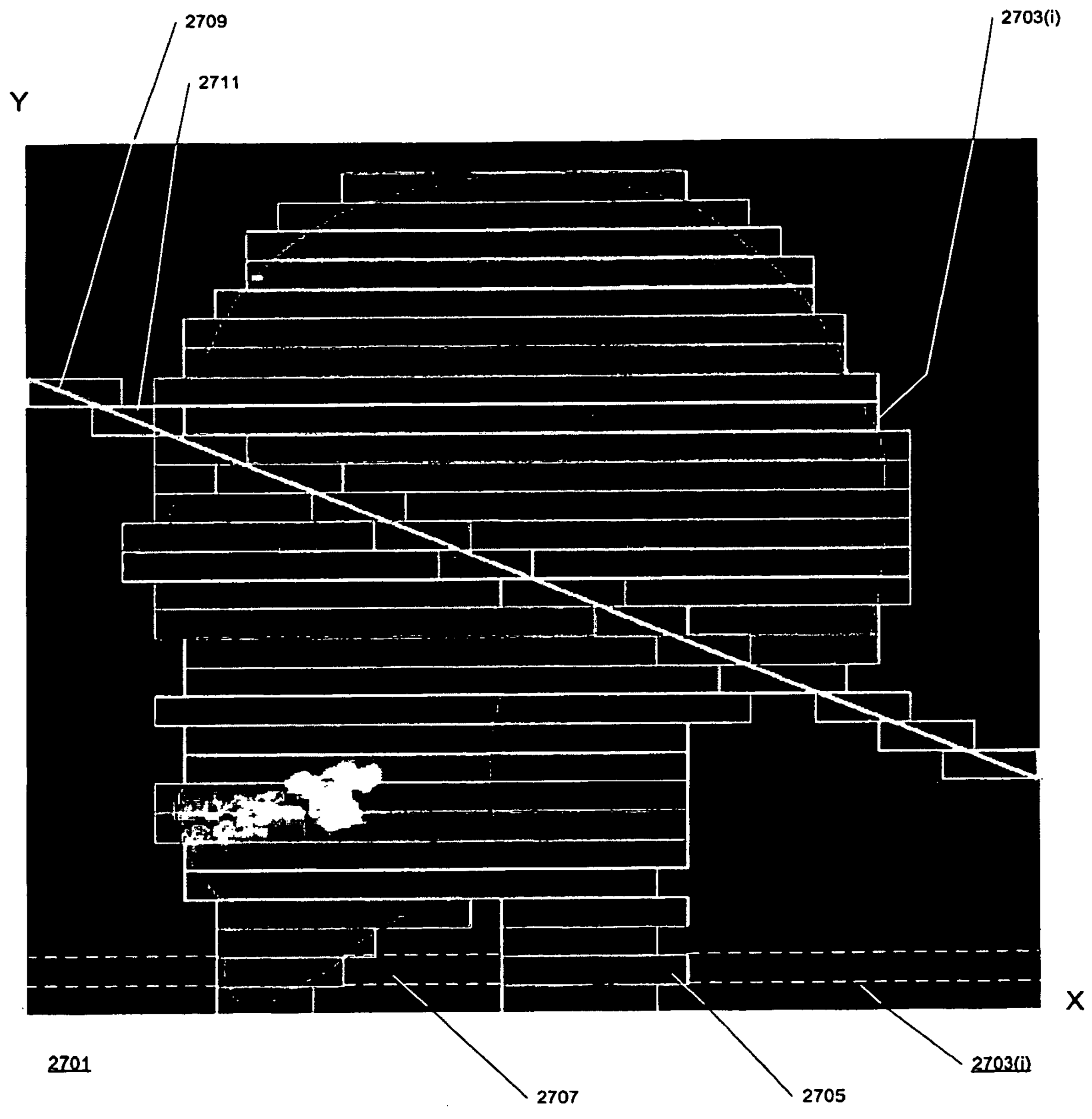


Fig. 27

```
For each run in the list
for $i = 0$; $i < list.length$; $i++$

    if $ray.run.end < list.run[i].start$
        No intersection exists
        return

    if $ray.run.start < list.run[i].end$
        Intersection exists
        $x_0 = \max(ray.run.start, list.run[i].start)$
        $x_1 = \min(ray.run.end, list.run[i].end)$
        $subdivision.traverseRun(x_1 - x_0, x_0, y, z)$
```

2801

Fig. 28

```
$i = 0$
$j = list.length$

if $ray.run.end < list.run[i].start$
    No intersection exists
    return

if $ray.run.start \geq list.run[j].end$
    No intersection exists
    return

$intersectRunList(i, j)$
```

```
$intersectRunList(int i, int j)$
    if $i == j$
        Intersection exists
        $x_0 = \max(ray.run.start, list.run[i].start)$
        $x_1 = \min(ray.run.end, list.run[i].end)$
        $subdivision.traverseRun(x_1 - x_0, x_0, y, z)$
        return

    $j' = \lfloor (i + j)/2 \rfloor$
    $i' = j' + 1$

    if $ray.run.start < list.run[j'].end$
        $intersectRunList(i, j')$

    if $ray.run.end \geq list.run[i'].start$
        $intersectRunList(i', j)$
```

2901

Fig. 29

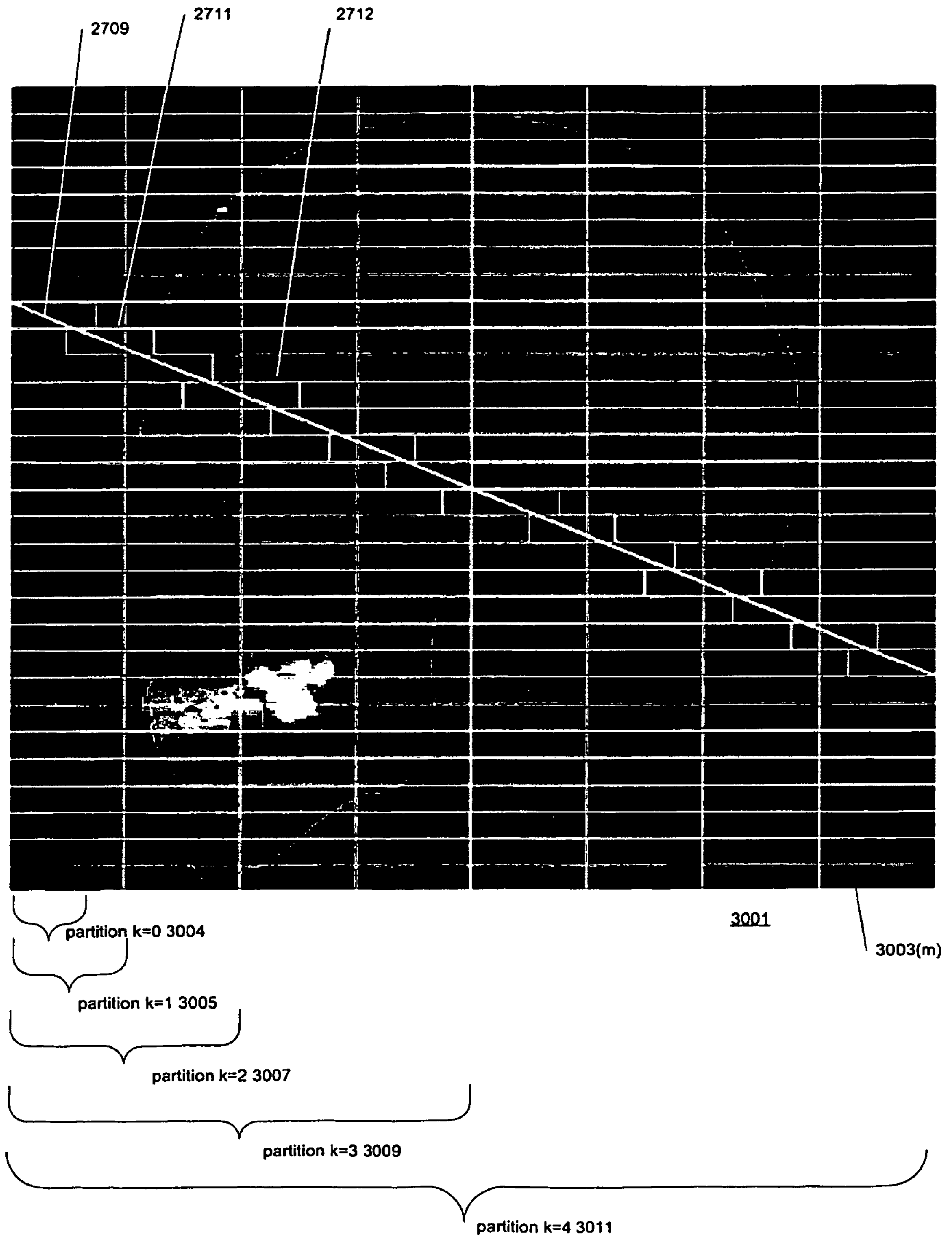

2709
2711
2712
3001
3003(m)
partition k=0 3004
partition k=1 3005
partition k=2 3007
partition k=3 3009
partition k=4 3011

Fig. 30

```
 Assume for each run:
$ray.run.length \leq partition.size$
for each run length
    if $ray.run.length > partition.extent$
        Handle tail of run length.
        if partition is interesting
            Traverse partition.extent cells.
        $ray.run.length -= partition.extent$
        $partition.extent = partition.size$
    Handle head of run length.
    if partition is interesting
        Traverse ray.run.length cells.
    $partition.extent -= ray.run.length$
```

3101

Fig. 31